

# Theoretical Analysis of the Optimal Free Responses of Graph-Based SFA for the Design of Training Graphs (Preprint)

Alberto N. Escalante-B.

Laurenz Wiskott

*Theory of Neural Systems*

*Institut für Neuroinformatik*

*Ruhr-University Bochum*

*Bochum D-44801, Germany*

ALBERTO.ESCALANTE@INI.RUB.DE

LAURENZ.WISKOTT@INI.RUB.DE

## Abstract

Slow feature analysis (SFA) is an unsupervised learning algorithm that extracts slowly varying features from a time series. Graph-based SFA (GSFA) is a *supervised* extension that can solve regression problems if followed by a post-processing regression algorithm. A training graph specifies arbitrary connections between the training samples. The connections in current graphs, however, only depend on the *rank* of the involved labels. Exploiting the exact label values makes further improvements in estimation accuracy possible.

In this article, we propose the exact label learning (ELL) method to create a graph that codes the desired label explicitly, so that GSFA is able to extract a normalized version of it directly. The ELL method is used for three tasks: (1) We estimate gender from artificial images of human faces (regression) and show the advantage of coding additional labels, particularly skin color. (2) We analyze two existing graphs for regression. (3) We extract *compact* discriminative features to classify traffic sign images. When the number of output features is limited, a higher classification rate is obtained compared to a graph equivalent to nonlinear Fisher discriminant analysis. The method is versatile, directly supports multiple labels, and provides higher accuracy compared to current graphs for the problems considered.

**Keywords:** Slow Feature Analysis, Nonlinear Regression, Image Analysis, Pattern Recognition, Many Classes

## 1. Introduction

The slowness principle is one of the learning paradigms that might explain, at least in part, the self-organization of neurons in the brain to extract invariant representations of relevant features. This principle operates on an abstract level and postulates that the most relevant abstract information that can be extracted from the environment typically changes much slower than the individual sensory inputs (e.g., the position of a bug compared to the quickly changing neural activations in the retina of a frog observing it).

The slowness principle was probably first formulated by Hinton (1989), and online learning rules were developed shortly after by Földiák (1991) and Mitchison (1991). The first closed-form algorithm is referred to as slow feature analysis (SFA, Wiskott, 1998; Wiskott and Sejnowski, 2002).

Recently, an extension of SFA for supervised learning called graph-based SFA (GSFA, Escalante-B. and Wiskott, 2013) has been proposed. In contrast to SFA, which is trained with a sequence of samples, GSFA is trained with a so-called training graph, in which the vertices are the samples and the edge weights represent similarities of the corresponding labels. In SFA, slowness requires the minimization of the squared output differences between (temporally) consecutive pairs of samples, whereas in GSFA the pairs of samples do not have to be consecutive, are weighted, and are defined by the training graph.

GSFA can be used to explicitly exploit the available labels by establishing an output similarity objective involving arbitrary samples. Typically, GSFA is more effective than SFA at extracting a set of features that tend to concentrate the label information and allow the accurate prediction of the labels, implicitly solving the supervised learning problem.

Although GSFA and locality preserving projections (LPP, He and Niyogi, 2003) originate from different backgrounds and were first motivated with different goals and applications in mind, there is a close relation between them, sharing very similar objective functions and constraints. Two differences are that in GSFA the vertex weights are independent of the edge weights and that GSFA is invariant to the scale of the weights, providing a normalized objective function. It is possible to use GSFA to compute LPP features, and vice versa. The results of this article might thus also be of interest for the LPP community.

In real life, many supervised learning problems are solved by applying feature extraction, (unsupervised) dimensionality reduction (DR), and an explicit supervised step (Figure 1.a). Another approach (Figure 1.b), based on GSFA, first uses GSFA for *supervised* DR, and then post-processes a small number of slow features with a conventional classification or regression algorithm. Supervised DR might result in higher accuracy than unsupervised DR. The supervised learning problem is mostly solved by GSFA implicitly, because it frequently concentrates the label-predictive information in a few features. Therefore, the post-processing step is not crucial, and might be a simple mapping from the slow features to the label domain.

Various training graphs for classification (clustered graph) and regression (e.g., serial, mixed, sliding window graphs) have been proposed (Escalante-B. and Wiskott, 2013). These pre-defined graphs offer great efficiency; although the number of edges contained in them is  $\mathcal{O}(N^2)$ , where  $N$  is the number of samples, their structure makes the training complexity linear w.r.t.  $N$ .

However, the construction of pre-defined graphs only takes into account the rank of the labels and not their exact value, a simplification that might decrease the estimation accuracy. In this article, we focus on the analysis and design of training graphs. We explore a new approach for solving regression problems with GSFA based on the construction of a special training graph, in which the slowest feature extracted is already a label estimation, up to a linear scaling (Figure 1.c). To develop this exact label learning (ELL) method, we first study the slowest possible features that can be extracted by GSFA from a given graph when the feature space is unrestricted. Such features have also been called optimal free responses and have been computed for SFA in continuous time by Wiskott (2003) using variational calculus. For GSFA, we use a different method based on linear algebra to cope with the discrete nature of the index  $n$  that takes the place of the time.

Expressing the optimal free responses of GSFA in closed form then allows us to develop a theoretical method for the converse operation; from a set of free responses we design

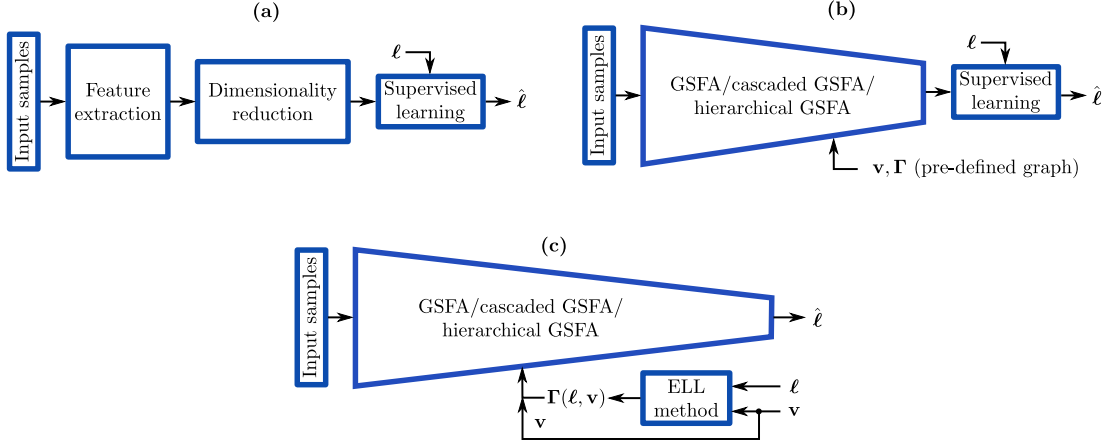


Figure 1: Three approaches for solving supervised learning problems. (a) A traditional and common approach. (b) Previous approach using GSFA with a pre-defined training graph, which is defined by the input samples, node weights  $\mathbf{v}$ , and edge weights  $\Gamma$ . The samples are assumed to be ordered by increasing label. (c) The approach proposed here, which consists of a single GSFA architecture that is trained with a specially constructed graph  $\Gamma(\ell, \mathbf{v})$ . The first slow feature (with a global sign adjustment) directly provides the label estimation. If the label  $\ell$  does not have weighted zero mean and weighted unit variance, a final linear scaling should be included.

the corresponding training graph. The method allows the creation of a graph in which the slowest possible feature is the label to be learned. Moreover, one can learn *multiple labels* simultaneously (e.g., object position, average color, shape, and size), and balance their importance. This property can be exploited to learn *auxiliary labels*, which provide a redundant coding of the original labels that may increase their estimation accuracy. In this case, as later explained, it is possible and desirable to emphasize the importance of the original labels over the auxiliary ones.

Cascaded GSFA refers to the consecutive application of multiple passes of GSFA. One advantage over direct GSFA is that the joint feature space may be more complex. Hierarchical GSFA (HGSFA), similarly to hierarchical SFA, is a divide-and-conquer approach for the extraction of slow features from high-dimensional data. HGSFA offers an excellent computational complexity compared to direct GSFA that can be as good as linear w.r.t the number of samples and the input dimensionality, depending on the network architecture. A graph designed with the proposed ELL method can be used to train GSFA, cascaded GSFA, and HGSFA, being thus also applicable to high-dimensional data.

Although the ELL method is based on theory and mostly contributes to a deeper understanding of GSFA, it can also be used in practice, providing higher accuracy than pre-defined graphs. While in general the ELL method results in a higher complexity compared to GSFA trained with an efficient pre-defined graph, it is still computationally viable for some datasets without resorting to specialized hardware or parallelization.

In the next section, we shortly review GSFA. In Section 3, we propose the ELL method. In Section 4, we provide three applications. Firstly, we solve a regression problem on gender estimation from artificial images, validating the method. Secondly, we analyze efficient pre-defined training graphs for regression. Thirdly, we use the ELL method in a different way to design a training graph for the extraction of compact features for classification, yielding improved performance when  $\log_2(C)$  to  $C - 2$  features are preserved. For the first and third application, the accuracy is evaluated experimentally. Section 5 closes the article with a discussion.

## 2. Graph-Based SFA (GSFA)

In this section, we recall the GSFA optimization problem, review the GSFA algorithm, and show how GSFA can be trained for both classification and regression.

### 2.1 Training Graphs and the GSFA Problem

GSFA is trained with a so-called training graph, in which the vertices are the samples and the edges between two samples may represent or be related to the similarity of their labels.

In mathematical terms, the training data is represented as a training graph  $G = (\mathbf{V}, \mathbf{E})$  (illustrated in Figure 2.a) with a set  $\mathbf{V} = \{\mathbf{x}(n)\}_n$  of vertices, each vertex being a sample, and a set  $\mathbf{E}$  of edges  $(\mathbf{x}(n), \mathbf{x}(n'))$ , which are pairs of samples, with  $1 \leq n, n' \leq N$ . The index  $n$  (or  $n'$ ) replaces the time variable  $t$  used by SFA. The edges are directed but typically have symmetric weights  $\mathbf{\Gamma}^T = \mathbf{\Gamma} = \{\gamma_{n,n'}\}_{n',n}$ ; weights  $v_n > 0$  are associated with the vertices  $\mathbf{x}(n)$  and can be used to reflect their importance, frequency, or reliability. This representation includes the standard time series of SFA as a special case in which the graph has a linear structure (see Figure 2.b).

Training graphs for classification typically favor connections between samples from the same class by means of larger edge weights compared to those of different classes, whereas training graphs for regression favor connections between samples with similar labels.

The concept of slowness has been generalized from sequences of samples (as in SFA) to training graphs. The general goal is to extract features that fulfill certain normalization restrictions and minimize the sum of the weighted squared output differences of all connected samples. More formally, the GSFA optimization problem (Escalante-B. and Wiskott, 2013) can be stated as follows. For  $1 \leq j \leq J$ , find features  $y_j(n) \stackrel{\text{def}}{=} g_j(\mathbf{x}(n))$ , where  $1 \leq n \leq N$  and  $g_j$  belongs to the feature space  $\mathcal{F}$  (frequent choices for  $\mathcal{F}$  are all linear or quadratic transformations of the inputs), such that the objective function (weighted delta value)

$$\Delta_j \stackrel{\text{def}}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} (y_j(n') - y_j(n))^2 \text{ is minimal} \quad (1)$$

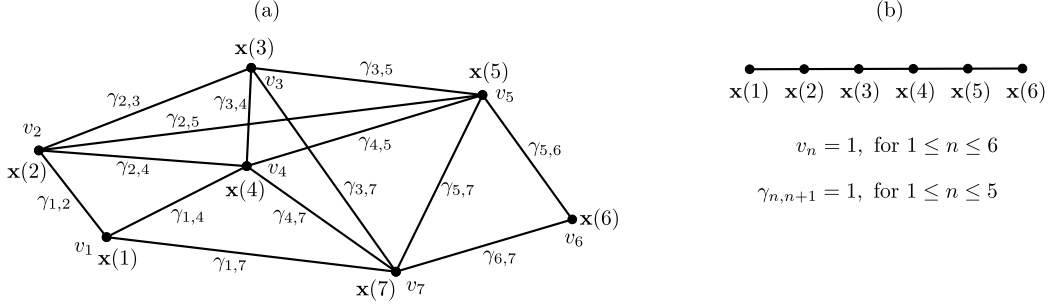


Figure 2: (a) Example of a training graph with  $N = 7$  vertices. (b) A regular sample sequence (time series), which can be used to train SFA. This sequence is represented here as a linear graph that can be used with GSFA. If labels are available and the samples have been reordered by increasing/decreasing label (e.g., instead of having been ordered by time), it is called *sample reordering* graph. (Figure from Escalante-B. and Wiskott, 2013).

under the constraints

$$\frac{1}{Q} \sum_n v_n y_j(n) = 0, \quad (2)$$

$$\frac{1}{Q} \sum_n v_n (y_j(n))^2 = 1, \text{ and} \quad (3)$$

$$\frac{1}{Q} \sum_n v_n y_j(n) y_{j'}(n) = 0, \text{ for } j' < j, \quad (4)$$

$$\text{with } Q \stackrel{\text{def}}{=} \sum_n v_n \text{ and } R \stackrel{\text{def}}{=} \sum_{n,n'} \gamma_{n,n'}. \quad (5)$$

The objective function penalizes the squared output differences between arbitrary pairs of samples using the edge weights as weighting factors. The feature  $y_1(n)$ , for  $1 \leq n \leq N$ , is the slowest one,  $y_2(n)$  is the second slowest, and so on. Constraints (2)–(4) are called weighted zero mean, weighted unit variance, and weighted decorrelation, respectively. They are similar to the normalization constraints of SFA, except for the inclusion of vertex weights. The factors  $1/R$  and  $1/Q$  are not essential for the optimization problem, but they provide invariance to the scale of the edge weights as well as to the scale of the vertex weights, and serve a normalization purpose.

## 2.2 Linear GSFA Algorithm

The linear GSFA algorithm is similar to standard SFA (Wiskott and Sejnowski, 2002) and only differs in the computation of the matrices  $\mathbf{C}$  and  $\dot{\mathbf{C}}$ , which in GSFA takes into account the neighborhood structure specified by the training graph (samples, edges, and weights).

The sample covariance matrix  $\mathbf{C}_\mathbf{G}$  is defined as:

$$\mathbf{C}_\mathbf{G} \stackrel{\text{def}}{=} \frac{1}{Q} \sum_n v_n (\mathbf{x}(n) - \tilde{\mathbf{x}})(\mathbf{x}(n) - \tilde{\mathbf{x}})^T, \quad (6)$$

where  $\mathbf{x}(n)$  and  $v_n$  denote an input sample and its weight, respectively, and

$$\tilde{\mathbf{x}} \stackrel{\text{def}}{=} \frac{1}{Q} \sum_n v_n \mathbf{x}(n) \quad (7)$$

is the weighted average of all samples. The derivative second-moment matrix  $\dot{\mathbf{C}}_\mathbf{G}$  is defined as:

$$\dot{\mathbf{C}}_\mathbf{G} \stackrel{\text{def}}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} (\mathbf{x}(n') - \mathbf{x}(n)) (\mathbf{x}(n') - \mathbf{x}(n))^T, \quad (8)$$

where edge weights  $\gamma_{n,n'}$  are defined as 0 if the graph does not have an edge  $(\mathbf{x}(n), \mathbf{x}(n'))$ .

Given these matrices, a sphering matrix  $\mathbf{S}$  and a rotation matrix  $\mathbf{R}$  are computed with

$$\mathbf{S}^T \mathbf{C}_\mathbf{G} \mathbf{S} = \mathbf{I}, \text{ and} \quad (9)$$

$$\mathbf{R}^T \mathbf{S}^T \dot{\mathbf{C}}_\mathbf{G} \mathbf{S} \mathbf{R} = \mathbf{\Lambda}, \quad (10)$$

where  $\mathbf{\Lambda}$  is a diagonal matrix with diagonal elements  $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_J$ . Finally the algorithm returns  $\Delta(y_1), \dots, \Delta(y_J)$ ,  $\mathbf{W}$  and  $\mathbf{y}(n)$ , where

$$\mathbf{W} = \mathbf{S} \mathbf{R}, \text{ and} \quad (11)$$

$$\mathbf{y}(n) = \mathbf{W}^T (\mathbf{x}(n) - \tilde{\mathbf{x}}). \quad (12)$$

It has been shown that the GSFA algorithm presented above indeed solves the optimization problem (1–4) in the linear function space. The proof is similar to the corresponding proof of standard linear SFA (Wiskott and Sejnowski, 2002).

**Probabilistic interpretation of a graph.** Interestingly, if the graph is connected and the following consistency restriction is fulfilled

$$\forall n : v_n/Q = \sum_{n'} \gamma_{n,n'}/R, \quad (13)$$

then GSFA yields the same features as standard SFA trained on a sequence generated by using the graph as a Markov chain with transition probabilities  $\gamma_{n,n'}/R$  (see Klampfl and Maass, 2010; Escalante-B. and Wiskott, 2013). Thus, one can use SFA to emulate GSFA. However, depending on the training graph used, emulating GSFA with SFA may be more computationally expensive.

### 2.3 Clustered and Serial Training Graphs

In this section, we present two efficient pre-defined graphs, the *clustered graph* for classification and the *serial graph* for regression (Escalante-B. and Wiskott, 2013).

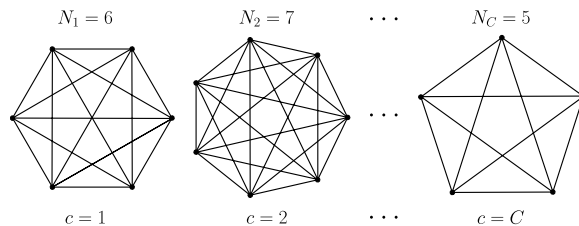


Figure 3: Illustration of a *clustered* training graph used for a classification problem with  $C$  classes. Each vertex represents a sample, and edges represent transitions. The  $N_c$  samples belonging to a class  $c \in \{1, \dots, C\}$  are connected, constituting a fully connected subgraph. Samples of different classes are not connected. Vertex weights are identical and equal to one, whereas edge weights depend on the cluster size as  $\gamma_{n,n'} = 1/(N_c - 1)$ , where  $\mathbf{x}(n)$  and  $\mathbf{x}(n')$  belong to class  $c$  and  $n \neq n'$ . (Figure from Escalante-B. and Wiskott, 2013).

The clustered graph, described in Figure 3, generates features useful for classification. The optimization problem associated with this graph explicitly demands that samples from the same class should typically be mapped to similar outputs.

The features learned by GSFA on this graph are equivalent to those learned by Fisher discriminant analysis (FDA, see Klampfl and Maass, 2010 and also compare Berkes, 2005a and Berkes, 2005b). This type of problem can be analyzed theoretically when the function space of SFA is unrestricted. Consistent with FDA, the first  $C - 1$  slow features extracted (optimal free responses) are orthogonal step functions, and are piece-wise constant for samples from the same class (Berkes, 2005a).

The *serial* training graph, described in Figure 4, is constructed by discretizing the original label  $\ell$  into a relatively small set of  $K$  discrete label values, namely  $\{\ell_1, \dots, \ell_K\}$ , where  $\ell_1 < \ell_2 < \dots < \ell_K$ . Afterwards, the samples are divided into  $K$  groups of size  $N/K$  sharing the same discrete labels. Edges connect all pairs of samples from consecutive groups with discrete labels  $\ell_k$  and  $\ell_{k+1}$ , for  $1 \leq k \leq K - 1$ . Thus, connections are only inter-group, and intra-group connections do not appear. Notice that since any two vertices of the same group are adjacent to exactly the same neighbors, they are likely to be mapped to similar outputs by GSFA. Following GSFA a complementary explicit regression step on a few features solves the original regression problem. There are several efficient graphs for regression besides the *serial graph*. We employ the serial graph in this article, because it has consistently given good results for various regression problems.

The clustered and serial graphs allow efficient training in linear time w.r.t  $N$ , whereas the number of connections considered is  $\mathcal{O}(N^2)$  if the number of clusters or groups is constant.

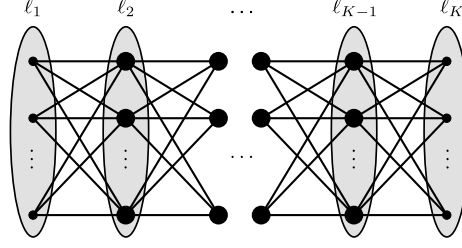


Figure 4: Illustration of a serial training graph with  $K$  discrete label values  $\{\ell_1, \dots, \ell_K\}$ . In this graph, the vertices are ordered and then grouped according to their discrete label. Even though the original label of two samples might differ, they will be grouped together if they have the same discrete label. Each dot represents a sample, edges represent connections, and ovals represent groups of samples. The samples of groups with discrete labels  $\ell_2$  to  $\ell_{K-1}$  have a weight of 2, whereas the samples of extreme groups with labels  $\ell_1$  or  $\ell_K$  have a weight of 1 (sample weights represented by bigger/smaller dots). The weight of all edges is 1. (Figure from Escalante-B. and Wiskott, 2013).

## 2.4 GSFA Optimization Problem in Matrix Notation

In order to apply linear algebra methods to analyze GSFA, we use matrix notation. In what follows we assume that the edge weights are symmetric<sup>1</sup> ( $\mathbf{\Gamma} = \mathbf{\Gamma}^T$ ) and that the consistency restriction (13) is fulfilled. This restriction can also be written as

$$\mathbf{v} \stackrel{(13)}{=} \frac{Q}{R} \mathbf{\Gamma} \mathbf{1}, \quad (14)$$

where  $\mathbf{1}$  is a vector of ones of length  $N$ .

If  $\mathbf{y}$  is a feasible solution (i.e., satisfying (2) and (3)) and the graph fulfills the consistency restriction (14), the weighted delta value (1) can be simplified as follows,

$$\Delta_{\mathbf{y}} \stackrel{(1)}{=} \frac{1}{R} \sum_{n,n'} \gamma_{n,n'} (y(n') - y(n))^2 \quad (15)$$

$$= \frac{1}{R} \left( \sum_{n'} (y(n'))^2 \sum_n \gamma_{n,n'} + \sum_n (y(n))^2 \sum_{n'} \gamma_{n,n'} - 2 \sum_{n,n'} \gamma_{n,n'} y(n') y(n) \right) \quad (16)$$

$$\stackrel{(14)}{=} \frac{1}{R} \left( \sum_{n'} (y(n'))^2 \frac{R}{Q} v(n') + \sum_n (y(n))^2 \frac{R}{Q} v(n) - 2 \mathbf{y}^T \mathbf{\Gamma} \mathbf{y} \right) \quad (17)$$

$$\stackrel{(3)}{=} 2 - \frac{2}{R} \mathbf{y}^T \mathbf{\Gamma} \mathbf{y}. \quad (18)$$

---

1. An asymmetric edge-weight matrix  $\mathbf{\Gamma}$  can be converted into a symmetric one  $\mathbf{\Gamma}' \stackrel{\text{def}}{=} \frac{\mathbf{\Gamma} + \mathbf{\Gamma}^T}{2}$  without altering the solution to the optimization problem.



The optimization problem can then be stated as: For  $1 \leq j \leq J$ , find vectors  $\mathbf{y}_j$  of length  $N$ , with  $y_j(n) \stackrel{\text{def}}{=} g_j(\mathbf{x}(n))$  and  $g_j \in \mathcal{F}$ , minimizing

$$\Delta_j \stackrel{(1,3,14)}{=} 2 - \frac{2}{R} \mathbf{y}_j^T \mathbf{\Gamma} \mathbf{y}_j \quad (19)$$

subject to:

$$\mathbf{v}^T \mathbf{y}_j \stackrel{(2)}{=} 0 \quad (20)$$

$$\mathbf{y}_j^T \text{Diag}(\mathbf{v}) \mathbf{y}_j \stackrel{(3)}{=} Q \quad (21)$$

$$\mathbf{y}_j^T \text{Diag}(\mathbf{v}) \mathbf{y}_{j'} \stackrel{(4)}{=} 0, \text{ for } j' < j, \quad (22)$$

where

$$Q \stackrel{(5.a)}{=} \mathbf{1}^T \mathbf{v}, \quad (23)$$

$$R \stackrel{(5.b)}{=} \mathbf{1}^T \mathbf{\Gamma} \mathbf{1}, \quad (24)$$

and  $\text{Diag}(\mathbf{v})$  denotes a diagonal matrix with diagonal  $\mathbf{v}$ .

### 3. Explicit Label Learning for Regression Problems

#### 3.1 Optimal Free Responses of GSFA

In this section, we calculate the slowest possible solutions (optimal free responses) to the GSFA problem (19)–(22) that one could find if the feature space were unlimited.

We use the Lagrange multiplier method to find critical points  $\mathbf{y}$  that are candidates for the optimal free responses. For the moment, we ignore the weighted decorrelation constraint (22) to solve for the first optimal free response, but we consider the remaining responses later. Due to the close relationship between GSFA and LPP, the approach below is strongly related to Laplacian Eigenmaps (Belkin and Niyogi, 2003). Let

$$L \stackrel{\text{def}}{=} \left(2 - \frac{2}{R} \mathbf{y}^T \mathbf{\Gamma} \mathbf{y}\right) + \alpha \mathbf{v}^T \mathbf{y} + \beta (\mathbf{y}^T \text{Diag}(\mathbf{v}) \mathbf{y} - Q) \quad (25)$$

be a Lagrangian corresponding to the objective function (19), under the constraints (20) and (21). A signal  $\mathbf{y}$  is a critical point if the partial derivatives of  $L$  with respect to  $\alpha, \beta$ , and  $y(n)$ , for  $1 \leq n \leq N$ , are simultaneously zero:

$$\partial L / \partial \alpha \stackrel{(25)}{=} \mathbf{v}^T \mathbf{y} \stackrel{!}{=} 0, \quad (26)$$

$$\partial L / \partial \beta \stackrel{(25)}{=} \mathbf{y}^T \text{Diag}(\mathbf{v}) \mathbf{y} - Q \stackrel{!}{=} 0, \text{ and} \quad (27)$$

$$\partial L / \partial \mathbf{y} \stackrel{(25)}{=} -\frac{4}{R} \mathbf{\Gamma} \mathbf{y} + \alpha \mathbf{v} + 2\beta \text{Diag}(\mathbf{v}) \mathbf{y} \stackrel{!}{=} \mathbf{0}, \quad (28)$$

where  $\mathbf{0}$  is a vector of zeros.

Equations (26) and (27) merely require that the output  $\mathbf{y}$  has weighted zero mean and weighted unit variance, respectively. Multiplying (28) with  $\mathbf{1}^T$  from the left and taking into account that  $\mathbf{1}^T \text{Diag}(\mathbf{v}) = \mathbf{v}^T$ ,  $\mathbf{1}^T \mathbf{v} \stackrel{(23)}{=} Q$ ,  $\mathbf{1}^T \mathbf{\Gamma} \stackrel{(14)}{=} \frac{R}{Q} \mathbf{v}^T$ , and  $Q > 0$  results in:

$$-\frac{4}{R} \left( \frac{R}{Q} \mathbf{v}^T \right) \mathbf{y} + \alpha Q + 2\beta \mathbf{v}^T \mathbf{y} = 0, \quad (29)$$

implying  $\alpha = 0$  due to (26). Therefore, (28) can be simplified to:

$$\left( -\frac{4}{R} \mathbf{\Gamma} + 2\beta \text{Diag}(\mathbf{v}) \right) \mathbf{y} = \mathbf{0}, \quad (30)$$

$$\Leftrightarrow \text{Diag}(\mathbf{v}^{-1/2}) \left( \frac{4}{R} \mathbf{\Gamma} - 2\beta \text{Diag}(\mathbf{v}) \right) \text{Diag}(\mathbf{v}^{-1/2}) \text{Diag}(\mathbf{v}^{1/2}) \mathbf{y} = \mathbf{0}, \quad (31)$$

$$\Leftrightarrow \left( \frac{4}{R} \text{Diag}(\mathbf{v}^{-1/2}) \mathbf{\Gamma} \text{Diag}(\mathbf{v}^{-1/2}) - 2\beta \mathbf{I} \right) (\text{Diag}(\mathbf{v}^{1/2}) \mathbf{y}) = \mathbf{0}, \quad (32)$$

$$\Leftrightarrow \left( \text{Diag}(\mathbf{v}^{-1/2}) \mathbf{\Gamma} \text{Diag}(\mathbf{v}^{-1/2}) - \frac{R\beta}{2} \mathbf{I} \right) (\text{Diag}(\mathbf{v}^{1/2}) \mathbf{y}) = \mathbf{0}, \quad (33)$$

where  $\mathbf{v}^{1/2}$  is defined as the element-wise square root of the elements of  $\mathbf{v}$ , and  $\mathbf{v}^{-1/2}$  is defined similarly (as usual, weights  $v_j$  are required to be strictly positive).

In a few words,  $\mathbf{y}$  is a critical point if it fulfills the weighted normalization conditions and the vector  $\text{Diag}(\mathbf{v}^{1/2}) \mathbf{y}$  is an eigenvector of the matrix  $\mathbf{M}$  defined as

$$\mathbf{M} \stackrel{\text{def}}{=} \text{Diag}(\mathbf{v}^{-1/2}) \mathbf{\Gamma} \text{Diag}(\mathbf{v}^{-1/2}). \quad (34)$$

The corresponding eigenvalue is denoted

$$\lambda = \frac{R\beta}{2}. \quad (35)$$

We denote the (orthogonal) eigenvectors of the matrix  $\mathbf{M}$  as  $\mathbf{u}_j$  with  $\mathbf{u}_j^T \mathbf{u}_j = 1$ . Each eigenvector gives rise to a critical point  $\mathbf{y}_j \stackrel{\text{def}}{=} Q^{1/2} \text{Diag}(\mathbf{v}^{-1/2}) \mathbf{u}_j$  if the weighted normalization conditions are also satisfied by  $\mathbf{y}_j$ . The slowest possible solution is the critical point  $\mathbf{y}_j$  with the smallest  $\Delta$ -value. As we show below, the  $\Delta$ -value of a critical point  $\mathbf{y}_j$  is directly related to the eigenvalue  $\lambda_j$  of the eigenvector  $\mathbf{u}_j = Q^{-1/2} \text{Diag}(\mathbf{v}^{1/2}) \mathbf{y}_j$  of  $\mathbf{M}$  and can be computed as follows.

$$\Delta_{\mathbf{y}_j} \stackrel{(19)}{=} 2 - \frac{2}{R} (\mathbf{y}_j)^T \mathbf{\Gamma} \mathbf{y}_j \quad (36)$$

$$\stackrel{(34)}{=} 2 - \frac{2}{R} (\mathbf{y}_j)^T \text{Diag}(\mathbf{v}^{1/2}) \mathbf{M} (\text{Diag}(\mathbf{v}^{1/2}) \mathbf{y}_j) \quad (37)$$

$$\stackrel{(35)}{=} 2 - \frac{2}{R} (\mathbf{y}_j)^T \text{Diag}(\mathbf{v}^{1/2}) \lambda_j \text{Diag}(\mathbf{v}^{1/2}) \mathbf{y}_j \quad (38)$$

$$\stackrel{(21)}{=} 2 - \frac{2Q}{R} \lambda_j \quad (39)$$

Thus, the slowest solution is the critical point  $\mathbf{y}_j$  with the largest eigenvalue  $\lambda_j$ . The remaining optimal free responses are the remaining critical points, and their the corresponding

eigenvalue defines their order, from largest to smallest. The weighted decorrelation condition (22) is fulfilled due to the orthogonality of the eigenvectors:  $\mathbf{u}_i^T \mathbf{u}_j = 0 \Leftrightarrow \frac{1}{Q} \mathbf{y}_i^T \text{Diag}(\mathbf{v}) \mathbf{y}_j = 0$  (follows from the definition of  $\mathbf{y}_j$  above).

One special case is when an eigenvalue has multiplicities. This means that two or more optimal free responses and any rotation of them have the same delta value. In this case, some optimal free responses are not unique, but any rotation of them is equivalent.

### 3.2 Design of a Training Graph for Learning One or Multiple Labels

Given a set of samples  $\{\mathbf{x}(1), \dots, \mathbf{x}(N)\}$  with label  $\ell = (\ell_1, \dots, \ell_N)$ , we show how to generate a training graph, such that the slowest feature that could be extracted by GSFA is equal to a normalized version of the label. Notice that this problem (determining the structure of a training graph, or more concretely, its edge-weight matrix  $\mathbf{\Gamma}$ , having a particular optimal solution) differs considerably from the original GSFA problem of finding an optimal solution given a training graph and a feature space. The approach can be extended to multiple labels per sample. To distinguish them, we introduce an index  $1 \leq j \leq L$ , making  $\ell_j$  denote the  $j$ -th label. In this case, the  $L$  labels can be expressed linearly in terms of the first  $L$  free responses.

Vertex-weights  $v_n$  indicate *a priori* likelihood information about the samples, and are thus assumed to be given and strictly positive. If this information is absent, one may set the vertex weights constant, e.g.  $\mathbf{v} = \frac{1}{N} \mathbf{1}$ .

Due to the normalization constraints, the outputs generated by GSFA must have weighted zero mean (20) and weighted unit variance (21). Therefore, we learn a weight-normalized label  $\tilde{\ell}$ , as follows: Let  $\mu_\ell = \frac{1}{Q} \mathbf{v}^T \ell$  be the weighted label average and  $\sigma_\ell^2 = \frac{1}{Q} (\ell - \mu_\ell \mathbf{1})^T \text{Diag}(\mathbf{v}) (\ell - \mu_\ell \mathbf{1})$  be the weighted label variance. Then, the normalized label is computed as

$$\tilde{\ell} = \frac{1}{\sigma_\ell} (\ell - \mu_\ell \mathbf{1}). \quad (40)$$

Hence, it is trivial to map from a normalized to a non-normalized label and *vice versa*.

In order for the construction to work when samples have multiple labels, we must weight decorrelate them first. To decorrelate two labels  $\ell_{j'}$  and  $\ell_j$ , with  $j' > j$ , one can project  $\ell_j$  out of  $\ell_{j'}$ ;  $\ell_{j'}^{\text{dec}}(n) = \ell_{j'}(n) - \frac{1}{Q} (\ell_{j'}^T \text{Diag}(\mathbf{v}) \ell_j) \ell_j(n)$ , which is an invertible linear operation.

From now on, we assume that the labels  $\ell_1, \dots, \ell_L$  are decorrelated and normalized. We compute edge weights  $\gamma_{n,n'}$  such that the  $j$ -th optimal free response is equal to  $\ell_j$  (with arbitrary polarity).

Define

$$\mathbf{\Gamma}^{\text{ELL}} \stackrel{\text{def}}{=} \text{Diag}(\mathbf{v}^{1/2}) \mathbf{M}^{\text{ELL}} \text{Diag}(\mathbf{v}^{1/2}), \quad (41)$$

where

$$\mathbf{M}^{\text{ELL}} \stackrel{\text{def}}{=} \sum_{j=0}^{N-1} \lambda_j \mathbf{u}_j^{\text{ELL}} (\mathbf{u}_j^{\text{ELL}})^T. \quad (42)$$

If  $L < N - 1$  one can set  $\lambda_{j>L} = 0$ . The matrix  $\mathbf{\Gamma}^{\text{ELL}}$  defined above is symmetric by construction. The eigenvectors and eigenvalues of  $\mathbf{M}^{\text{ELL}}$ , which are explicit in its eigenvector decomposition above, directly define the matrix  $\mathbf{\Gamma}^{\text{ELL}}$  and determine the optimal free

responses of the resulting graph. Concretely, for each  $j \geq 1$  one sets  $\mathbf{u}_j^{\text{ELL}}$  according to the desired label  $\ell_j$  (ignore  $\mathbf{u}_0^{\text{ELL}}$  and  $\lambda_0$  for the time being).

$$\mathbf{u}_j^{\text{ELL}} = Q^{-1/2} \text{Diag}(\mathbf{v}^{1/2}) \ell_j, \text{ for } j \geq 1 \quad (43)$$

Notice that the weighted decorrelation of the labels translates directly into the orthogonality of the corresponding eigenvectors

$$\frac{1}{Q} (\ell_j)^T \text{Diag}(\mathbf{v}) \ell_{j'} \stackrel{(22)}{=} 0 \stackrel{(43)}{\Leftrightarrow} (\mathbf{u}_j^{\text{ELL}})^T \mathbf{u}_{j'}^{\text{ELL}} = 0 \quad (44)$$

Once the eigenvectors are computed we must decide which eigenvalues we want to give them. Alternatively, we can decide which  $\Delta$  values we give to the labels, because  $\Delta_{\ell_j}$  and  $\lambda_j$  are directly related:  $\lambda_j \stackrel{(39)}{=} \frac{R}{2Q} (2 - \Delta_{\ell_j})$ .

Larger eigenvalues (equivalent to smaller  $\Delta$  values) might result in higher accuracy for the corresponding label. We give some intuition on how to choose the eigenvalues of the eigenvectors. a) In general, important labels should have larger eigenvalues than less important ones. b) The global scale of the eigenvalues  $\lambda_{j>0}$  is irrelevant, only their relative scales matter. For convenience one can scale them so that  $\sum \lambda_{j>0} = 1$ . c) If two labels are similarly important, their eigenvalues should be also similar.

For example, if one only wants to learn a single label  $\ell_1$  with a delta value  $\Delta_{\ell_1} = 0$ , one can set  $\mathbf{u}_1^{\text{ELL}} = Q^{-1/2} \text{Diag}(\mathbf{v}^{1/2}) \ell_1$ ,  $\lambda_1 = 1$ , and the eigenvalues  $\lambda_{j>1}$  to zero. If  $\ell_1$  takes only two possible values (e.g.,  $-1$  and  $1$ ), the resulting graph will be disconnected and contain two clusters. Otherwise, the resulting graph will be connected, and the condition  $\Delta_{\ell_1} = 0$  necessarily implies that some of the resulting edge weights will be negative, a condition that we deal with in Section 3.3.

The analysis of Section 3.1, which is used by the ELL method requires that the graph fulfills the consistency restriction (14), which depends on  $\mathbf{u}_0$  and  $\lambda_0$ , as follows,

$$\mathbf{\Gamma}^{\text{ELL}} \mathbf{1} \stackrel{(41,42)}{=} \text{Diag}(\mathbf{v}^{1/2}) \left( \sum \lambda_j \mathbf{u}_j^{\text{ELL}} (\mathbf{u}_j^{\text{ELL}})^T \right) \text{Diag}(\mathbf{v}^{1/2}) \mathbf{1} \quad (45)$$

$$= \text{Diag}(\mathbf{v}^{1/2}) \left( \sum \lambda_j \mathbf{u}_j^{\text{ELL}} (\mathbf{u}_j^{\text{ELL}})^T \right) \mathbf{u}_0^{\text{ELL}} Q^{1/2} \quad (46)$$

$$= \text{Diag}(\mathbf{v}^{1/2}) \lambda_0 \mathbf{u}_0^{\text{ELL}} Q^{1/2} \quad (47)$$

$$\stackrel{!}{=} (R/Q) \mathbf{v}. \quad (48)$$

To satisfy the equation above one can set  $\mathbf{u}_0^{\text{ELL}} = Q^{-1/2} \mathbf{v}^{1/2}$  with eigenvalue  $\lambda_0 = R/Q$ , which also ensures  $(\mathbf{u}_0^{\text{ELL}})^T \mathbf{u}_0^{\text{ELL}} = 1$  and  $\mathbf{1}^T \mathbf{\Gamma}^{\text{ELL}} \mathbf{1} = R$ . The free pseudo-response  $\ell_0 \stackrel{(43)}{=} \mathbf{1}$  corresponding to  $\mathbf{u}_0^{\text{ELL}}$  fulfills equations (21) and (22) but not (20). Therefore,  $\ell_0$  is not a feasible solution, but it has similar properties to the optimal free responses. The introduction of  $\mathbf{u}_0^{\text{ELL}}$  does not reduce the generality of the labels  $\ell_{j>0}$  that can be learned; orthogonality between  $\mathbf{u}_0^{\text{ELL}}$  and  $\mathbf{u}_{j>0}^{\text{ELL}}$  is equivalent to the weighted zero mean of  $\ell_{j>0}$  (20), which is required anyway for any feasible solution, i.e.,  $(\mathbf{u}_0^{\text{ELL}})^T \mathbf{u}_j^{\text{ELL}} = 0 \stackrel{(44)}{\Leftrightarrow} (Q^{-1/2} \mathbf{v}^{1/2})^T Q^{-1/2} \text{Diag}(\mathbf{v}^{1/2}) \ell_j = Q^{-1} \mathbf{v}^T \ell_j = 0$ .

Although only  $L$  free responses are explicitly defined,  $N - L - 1$  additional optimal free responses are defined implicitly with an eigenvalue of 0, corresponding to  $\Delta = 2.0$ . This

$\Delta$  value has a particular meaning, because as we prove in the next paragraph, it is the  $\Delta$  value of unit-variance zero-mean i.i.d. noise for certain graphs.

### 3.2.1 EXPECTED WEIGHTED $\Delta$ VALUE OF A NOISE FEATURE

Let  $\mathbf{y}$  be a noise feature randomly sampled from a zero-mean unit-variance distribution  $\mathcal{D}$ , i.e.,  $y(n) \leftarrow \mathcal{D}(0, 1)$ . On average,  $\mathbf{y}$  fulfills the normalization conditions, as can be seen as follows.

$$(20): \quad \langle \mathbf{v}^T \mathbf{y} \rangle_{\mathcal{D}} = \mathbf{v}^T \langle \mathbf{y} \rangle_{\mathcal{D}} = 0, \quad (49)$$

$$(21): \quad \langle \mathbf{y}^T \text{Diag}(\mathbf{v}) \mathbf{y} \rangle_{\mathcal{D}} = \langle \sum_n v_n y(n)^2 \rangle_{\mathcal{D}} = \sum_n v_n \langle y(n)^2 \rangle_{\mathcal{D}} = Q, \quad (50)$$

where  $\langle \cdot \rangle_{\mathcal{D}}$  denotes expected value when sampling over  $\mathcal{D}$ . The expected delta value can be computed as

$$\langle \Delta_{\mathbf{y}} \rangle_{\mathcal{D}} \stackrel{(1)}{=} \frac{1}{R} \sum_{n, n'} \gamma_{n, n'} \langle (y(n') - y(n))^2 \rangle_{\mathcal{D}} \quad (51)$$

$$= \frac{1}{R} \left( \sum_{n, n', n \neq n'} \gamma_{n, n'} \langle (y(n') - y(n))^2 \rangle_{\mathcal{D}} + \sum_n \gamma_{n, n} \langle (y(n) - y(n))^2 \rangle_{\mathcal{D}} \right) \quad (52)$$

$$= \frac{1}{R} \left( \sum_{n, n', n \neq n'} \gamma_{n, n'} (\langle y(n')^2 \rangle_{\mathcal{D}} + \langle y(n)^2 \rangle_{\mathcal{D}} - 2 \langle y(n') \rangle_{\mathcal{D}} \langle y(n) \rangle_{\mathcal{D}}) + 0 \right) \quad (53)$$

$$= \frac{1}{R} \sum_{n, n', n \neq n'} \gamma_{n, n'} (1 + 1 - 0) \quad (54)$$

$$= \frac{2}{R} \left( \sum_{n, n'} \gamma_{n, n'} - \sum_n \gamma_{n, n} \right) = \frac{2(R - \sum_n \gamma_{n, n})}{R}. \quad (55)$$

Therefore, if the graph has no self-loops (i.e.,  $\forall n : \gamma_{n, n} = 0$ ), the expected  $\Delta$  value  $\langle \Delta_{\mathbf{y}} \rangle_{\mathcal{D}}$  of a noise feature is 2. The self-loops of a graph (e.g., one constructed using the ELL method) can be removed without changing the free responses, only affecting the scale of the delta values due to the change in  $R$ . The consistency restriction might be broken, though.

### 3.3 Elimination of Negative Edge Weights

From the objective function (1), it is obvious that a positive edge weight connecting two samples expresses that those samples should be mapped close to each other in feature space. In contrast, negative edge weights express that two samples should be mapped as far apart as possible, thus encoding output dissimilarities. Nevertheless, the weighted unit variance constraint still applies, so the solutions are not unbounded.

If the edge weights are non-negative, the smallest possible  $\Delta$  value is  $\Delta = 0$ . However, if negative edge weights are allowed, some feasible features might have  $\Delta < 0$ . A feature with  $\Delta < 0$  would appear to be “slower” than the infeasible constant feature  $\mathbf{y} = \mathbf{1}$  with  $\Delta = 0$ , contradicting the intuitive interpretation of slowness. Moreover, negative edge

weights hinder the probabilistic interpretation of the graph (see Section 2.2), because some of the transition probabilities  $\gamma_{n,n'}/R$  of the resulting Markov chain would be negative.

Training graphs constructed using the ELL method might include negative edge weights, which would result in the disadvantages described above.

In this section, we add an additional step to the ELL method to ensure that the training graph has non-negative edge weights. More concretely, we show how to transform a training graph with strictly positive vertex weights  $v_n$  and arbitrary edge weights  $\mathbf{\Gamma}$  (positive and negative) into a graph with the same vertex weights and only non-negative edge weights  $\mathbf{\Gamma}'$ . The optimization problem defined by  $\mathbf{\Gamma}'$  is equivalent to the original optimization problem in terms of its solutions and their order. Only the value of the objective function is linearly changed (or, more precisely, changed by an affine function).

Assume that  $\forall n : v_n > 0$ , and that there is at least one element  $\gamma_{n,n'} < 0$ . Let  $c \stackrel{\text{def}}{=} \max_{n,n'} \frac{-\gamma_{n,n'}}{v_n v_{n'}}$ . The new edge weights  $\mathbf{\Gamma}'$  are defined as

$$\mathbf{\Gamma}' \stackrel{\text{def}}{=} \frac{1}{1 + cQ^2/R} (\mathbf{\Gamma} + c\mathbf{v}\mathbf{v}^T). \quad (56)$$

Now, we show the properties of  $\mathbf{\Gamma}'$  and its relation to  $\mathbf{\Gamma}$ :

1. All elements of  $\mathbf{\Gamma}'$  are greater or equal to zero, as desired. (Follows from the definition of  $c$ ;  $\gamma_{n,n'} + cv_n v_{n'} \geq 0$ .)
2. Symmetry is preserved; clearly  $\mathbf{\Gamma}'$  is symmetric if and only if  $\mathbf{\Gamma}$  is symmetric.
3. The sum of edge-weights is preserved:

$$R' = \mathbf{1}^T \mathbf{\Gamma}' \mathbf{1} = \frac{R + cQ^2}{1 + cQ^2/R} = R. \quad (57)$$

4. Fulfillment of the graph consistency restriction (14) is preserved:

$$\mathbf{1}^T \mathbf{\Gamma} \stackrel{(14)}{=} R/Q\mathbf{v}^T \quad \Rightarrow \quad \mathbf{1}^T \mathbf{\Gamma}' \stackrel{(56)}{=} \frac{1}{1 + cQ^2/R} (\mathbf{1}^T \mathbf{\Gamma} + c\mathbf{1}^T \mathbf{v}\mathbf{v}^T) \quad (58)$$

$$\stackrel{(14)}{=} \frac{1}{1 + cQ^2/R} (R/Q\mathbf{v}^T + cQ\mathbf{v}^T) \quad (59)$$

$$= \frac{R/Q}{1 + cQ^2/R} (1 + cQ^2/R) \mathbf{v}^T \quad (60)$$

$$= R/Q\mathbf{v}^T. \quad (61)$$

5.  $\mathbf{\Gamma}$  and  $\mathbf{\Gamma}'$  define equivalent optimization problems. Let  $\mathbf{y}$  be a feasible solution. The constraints of the optimization problem are independent of  $\mathbf{\Gamma}'$ , and only the objective

function is modified as follows:

$$\Delta'_y \stackrel{(18)}{=} 2 - \frac{2}{R'} \mathbf{y}^T \mathbf{\Gamma}' \mathbf{y} \quad (62)$$

$$\stackrel{(56,57)}{=} 2 - \frac{2}{R(1 + cQ^2/R)} (\mathbf{y}^T \mathbf{\Gamma} \mathbf{y} + c \mathbf{y}^T \mathbf{v} \mathbf{v}^T \mathbf{y}) \quad (63)$$

$$\stackrel{(20)}{=} 2 - \frac{2}{R(1 + cQ^2/R)} \mathbf{y}^T \mathbf{\Gamma} \mathbf{y} \quad (64)$$

$$\stackrel{(18)}{=} 2 - \frac{2}{R(1 + cQ^2/R)} \frac{R}{2} (2 - \Delta_y) \quad (65)$$

$$= \frac{1}{(1 + cQ^2/R)} \left( \Delta_y + \frac{2cQ^2}{R} \right). \quad (66)$$

Therefore, the objective function is only modified by a positive scaling factor and a constant positive offset, proving that the optimal free solutions to the training graph remain stable, as well as their order.

6. In particular, a feature  $\mathbf{y}$  with  $\Delta_y = 2$  preserves its delta value, i.e.  $\Delta_y = 2 \stackrel{(65)}{\Leftrightarrow} \Delta'_y = 2$ .

### 3.4 Auxiliary Labels for Boosting Estimation Accuracy

When GSFA is applied repeatedly (e.g., cascaded or in a convergent hierarchical GSFA network) one can provide additional *auxiliary* labels derived from the original one  $\ell_1$  to improve the estimation accuracy. Informally, even though a given GSFA node might not be able to extract  $\ell_1$  accurately, it might be capable of approximating features  $f_2(\ell_1), f_3(\ell_1), \dots, f_K(\ell_1)$ . Since these features are derived from the label, they contain information that (partially) determines it. Hence, a posterior GSFA node might be able to disentangle these features more effectively to recover the original label. One can explicitly promote the appearance of these features by learning also auxiliary labels  $\ell_k = f_k(\ell_1)$ , for  $2 \leq k \leq K$ .

The functions  $f_k$  can be defined arbitrarily, one simple choice is to use

$$\ell_k(n) = \cos \left( \frac{\ell_1(n) - \min(\ell_1)}{\max(\ell_1) - \min(\ell_1)} \pi k \right), \text{ for } 2 \leq k \leq K, \quad (67)$$

where  $\max(\ell_1)$  is the largest label value, and  $\min(\ell_1)$  is the smallest one. Notice that for  $k = 2$  the argument of the cosine function ranges from 0 to  $\pi$ , for  $k = 3$  from 0 to  $(3/2)\pi$ , etc. In this sense, these features are “higher-frequency” versions of  $\ell_1$ . The eigenvalues corresponding to the auxiliary labels must be set smaller than those of the original label. Otherwise, the slowest features found might be close to the auxiliary labels rather than to the original one. From now on, we use the term *target* labels to refer to the original and auxiliary labels, if present.

Interestingly, in regular SFA (or GSFA trained with the reordering graph) the inclusion of auxiliary labels occurs automatically. The slowest free response is a half period of a cosine function, and the subsequent free responses are the higher-frequency harmonics of the first one (see Section 4.2, particularly Figure 7).

### 3.5 Computational Complexity of Explicit Label Learning

The main drawback of ELL is its computational efficiency compared to efficient pre-defined training graphs, which is more marked for large  $N$ . We analyse the efficiency of explicit label learning by considering its two main parts: The construction of the training graph and training GSFA with it.

The graph construction requires  $\mathcal{O}(L^2N + LN^2)$  operations. The term  $L^2N$  is due to the transformation of  $L$  target labels into eigenvectors, which might require a decorrelation step on  $L$   $N$ -dimensional vectors. The term  $LN^2$  is due to the computation of  $\mathbf{M}$ , which involves  $L$  vector multiplications  $\mathbf{u}_j\mathbf{u}_j^T$ .

When training GSFA, three computations are particularly expensive. Firstly, the computation of  $\mathbf{C}_\mathbf{G}$ , which takes  $\mathcal{O}(NI^2)$  operations. Secondly, the computation of  $\dot{\mathbf{C}}_\mathbf{G}$ , which can be expressed as  $\dot{\mathbf{C}}_\mathbf{G} = \frac{2}{Q}\mathbf{X}\text{Diag}(\mathbf{v})\mathbf{X}^T - \frac{2}{R}\mathbf{X}\mathbf{\Gamma}\mathbf{X}^T$ , where  $\mathbf{X} = (\mathbf{x}_1, \dots, \mathbf{x}_N)$ , taking  $\mathcal{O}(N^2I + NI^2)$  operations. Thirdly, the solution to the generalized eigenvalue problem, which requires  $\mathcal{O}(I^3)$  operations. Therefore, in general, training GSFA requires  $\mathcal{O}(NI^2 + N^2I + I^3)$  operations. Typically  $N > I$  to avoid overfitting, so the computation of  $\dot{\mathbf{C}}_\mathbf{G}$  is the most expensive part.

However, when a pre-defined graph is used instead of an ELL graph,  $\dot{\mathbf{C}}_\mathbf{G}$  might be computed more efficiently by using optimized algorithms. If an efficient pre-defined graph is used (e.g. the serial graph),  $\dot{\mathbf{C}}_\mathbf{G}$  can be computed in  $\mathcal{O}(NI^2)$  operations (due to the regular structure of the graph), which is the complexity of the same operation using standard SFA on  $N$   $I$ -dimensional samples. Moreover, if the number of edges  $N_e \leq N(N+1)/2$  is small, one can use (8) to compute  $\dot{\mathbf{C}}_\mathbf{G}$  in  $\mathcal{O}(N_eI^2)$  operations. Therefore, for these two special cases, training GSFA takes  $\mathcal{O}(NI^2 + I^3)$  and  $\mathcal{O}((N_e + N)I^2 + I^3)$  operations, respectively.

## 4. Applications of Explicit Label Learning

In this section, we present three applications of the proposed method. First, we illustrate how to solve a regression problem with GSFA explicitly, learning a direct mapping from images to their labels (see Figure 1.c). In the second application, we analyse two pre-defined graphs by computing their optimal free responses. In the third application, we use the ELL method in a new way to learn compact discriminative labels for classification.

### 4.1 Explicit Estimation of Gender with GSFA

We consider the problem of gender estimation from artificial face images, which is treated here as a regression problem, because the gender parameter is defined as a real value by the face modelling software.

**Input data.** The input data are 20,000  $64 \times 64$  grayscale images. Each image is generated using a new subject identity, where the gender is specified by us, and the rest of the parameters of the faces (e.g., age, racial composition) are random. The average pixel intensity of each image is normalized by multiplying it by an appropriate factor. The resulting images show subjects with a fixed pose, no hair or accessories, fixed illumination, constant average pixel intensity, and a black background. See Figure 5 for some sample images. 60 different values of the gender parameter are used  $(-3, -2.9, \dots, 2.9)$ .



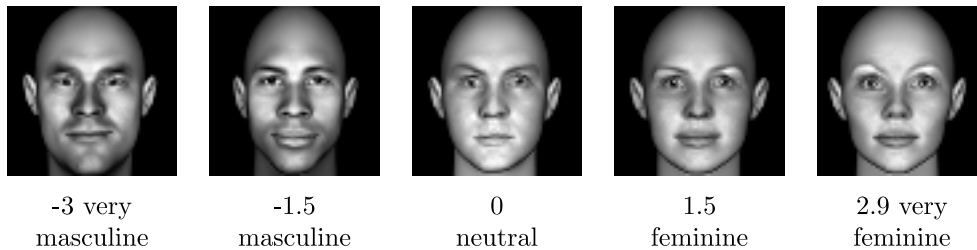


Figure 5: Example of the images used, showing different values of the gender parameter.

The images are randomly split into a training and a test set. The training set consists of 10,800 images, 180 images for each gender value, whereas the test set consists of 1,200 images, 20 images for each gender value.

Besides the gender label, we also consider a second “color” label, which is the average pixel intensity of the image *before* normalization. Due to normalization, this label cannot be computed directly, but it can be estimated from other cues, such as the subject’s apparent race and face size. In the following experiment we consider only the gender label, but later we use both labels (gender and color) simultaneously.

**Network used.** For efficiency reasons, hierarchical GSFA (HGSFA) is used. We test an 8-layer HGSFA network with the structure described in Table 1. The nodes of the network have non-overlapping receptive fields and are composed of an expansion function  $(x_1, \dots, x_n) \mapsto (x_1, \dots, x_n, |x_1|^{0.8}, \dots, |x_n|^{0.8})$  followed by linear GSFA. This expansion only doubles the data dimensionality and is called 0.8Expo (Escalante-B. and Wiskott, 2011). The nodes of the first layer include a PCA pre-processing step, in which 50 out of 64 components are preserved.

layer	number of nodes	node’s receptive field (pixels)	input dim per node	expanded dim per node	output dim per node
1	8×8	8×8	64	100	40
2	4×8	16×8	80	160	40
3	4×4	16×16	80	160	40
4	2×4	32×16	80	160	40
5	2×2	32×32	80	160	40
6	1×2	64×32	80	160	40
7	1×1	64×64	80	160	40
8	1×1	64×64	40	80	6

Table 1: Structure of the GSFA hierarchical network. The inputs to the nodes in the first layer are 8×8-pixel patches. The input to the node in layer 8 is the output of the node in layer 7. The inputs to all other nodes come from two contiguous (either vertically or horizontally) nodes in the previous layer.

**Training graphs for gender estimation.** Several training graphs are constructed with the ELL method described in Sections 3.2–3.4. The graphs are denoted  $\text{ELL}^g\text{-}L$ , where  $L$  is

the total number of target labels considered, with  $L \in \{1, 10, 20, 30, 40, 50\}$ . The first target label  $\ell_1(n)$  is the gender parameter, where  $1 \leq n \leq 10,800$ . The remaining  $L - 1$  labels are auxiliary and computed using (67). For comparison purposes, the serial and reordering training graphs were also tested.

**Label estimations.** We used three mappings from the slowest features to the label estimation  $\hat{\ell}$ . The first mapping (only available for the ELL graphs) is a linear scaling  $\hat{\ell} = \pm y_1 \sigma_\ell + \mu_\ell$  that inverts the label normalization (40). Since the sign of  $y_1$  is arbitrary, it is globally adjusted to best fit the labels. The second method is linear regression (LR). For these two methods, final label estimation  $\hat{\ell}$  is clipped to the valid label range  $[-3, 2.9]$ . The third mapping is the Soft GC method, which provides a soft estimation based on the class probabilities estimated by a Gaussian classifier (trained with 60 classes, Escalante-B. and Wiskott, 2013).

**Results.** Table 2 (left) shows the label estimation errors (RMSE) for the gender label. Depending on the mapping, the  $\text{ELL}^g$ -10 and  $\text{ELL}^g$ -40 graphs outperform the rest. This supports the intuition that auxiliary labels are useful. 50 target labels perform worse than 40, probably in part because the output dimensionality of the intermediate nodes in the network is 40. Without the final clipping step LR was clearly more accurate than linear scaling (experiment not shown), but both methods have similar accuracy if clipping is enabled. For all graphs, the explicitly supervised soft GC method provided better accuracy than the linear scaling method, although the difference is less than one might have expected.

Graph $\text{ELL}^g$ - $L$					Graph $\text{ELL}^{g,c}$ - $L$				
$L$	scaling (1F)	LR (1F)	soft GC (1F)	soft GC (3F)	$L$	scaling (1F)	LR (1F)	soft GC (1F)	soft GC (3F)
1	0.376	0.380	0.364	0.365	$2 \times 1$	<b>0.298</b>	<b>0.299</b>	<b>0.289</b>	0.284
10	<b>0.364</b>	<b>0.365</b>	0.353	0.356	$2 \times 5$	0.349	0.350	0.343	<b>0.277</b>
20	0.372	0.374	0.356	0.357	$2 \times 10$	0.423	0.426	0.410	0.288
30	0.367	0.368	0.350	0.349	$2 \times 15$	0.473	0.478	0.453	0.291
40	0.368	0.367	<b>0.346</b>	<b>0.345</b>	$2 \times 20$	0.508	0.514	0.479	0.292
50	0.376	0.375	0.351	0.350	$2 \times 25$	0.535	0.543	0.499	0.294

Table 2: Gender estimation errors (RMSE) using various graphs and either one (1F) or three (3F) features. For the linear regression (LR) mapping, the label is estimated as  $\hat{\ell}_1 = ay_1 + b$ , with  $a$  and  $b$  fitted to the training data. Chance level (RMSE) is 1.731 if one uses the constant estimation  $\hat{\ell}_1 = -0.05$ . All results on test data and averaged over 10 runs. (Left) Error using training graphs for gender estimation only. (Right) Error using training graphs for the experiment on simultaneous gender and color estimation.

For comparison, the serial graph results in RMSEs of 0.351 (soft GC, 1F) and 0.349 (soft GC, 3F), whereas the reordering graph results in RMSEs of 0.353 (soft GC, 1F) and 0.347 (soft GC, 3F). The accuracy of these two graphs appears to be similar; however, in more complex experiments the serial graph has typically been more accurate (e.g., Escalante-B. and Wiskott, 2013). The  $\text{ELL}^g$ -40 graph is, therefore, slightly more accurate than the serial

and reordering graphs but 25 times slower, taking about 250 min for training instead of about 10 min (single thread).

**Simultaneous learning of gender and color.** We construct a graph that codes gender and color simultaneously, learning labels  $\mathbf{y}_1, \dots, \mathbf{y}_L$ , where  $\mathbf{y}_1, \mathbf{y}_3, \dots, \mathbf{y}_{L-1}$  are derived from the gender label, and  $\mathbf{y}_2, \mathbf{y}_4, \dots, \mathbf{y}_L$  are derived from the color label. These labels are computed similarly to when learning gender only but using two different original labels. We use linearly decreasing eigenvalues. The resulting graphs are denoted  $\text{ELL}^{g,c}\text{-}L$ , where  $L$  is the total number of target labels, with  $L = 2 \times d$ , and  $2(d - 1)$  is the number of auxiliary labels used for gender and color.

Graph $\text{ELL}^c\text{-}L$					Graph $\text{ELL}^{g,c}\text{-}L$				
$L$	scaling (1F)	LR (1F)	soft GC (1F)	soft GC (3F)	$L$	LR (1F)	soft GC (1F)	LR (3F)	soft GC (3F)
1	2	1.987	1.971	1.979	$2 \times 1$	4.247	4.291	1.393	1.221
10	<b>1.969</b>	<b>1.958</b>	1.905	1.922	$2 \times 5$	3.606	3.614	<b>1.239</b>	1.210
20	2.006	1.999	1.914	1.922	$2 \times 10$	3.214	3.185	1.337	1.180
30	1.991	1.989	1.877	1.889	$2 \times 15$	2.978	2.945	1.429	1.158
40	1.990	1.990	<b>1.864</b>	<b>1.867</b>	$2 \times 20$	2.828	2.802	1.501	1.141
50	1.997	1.997	1.865	1.871	$2 \times 25$	<b>2.718</b>	<b>2.700</b>	1.582	<b>1.140</b>

Table 3: *Color* estimation errors (RMSE) using various graphs and either one (1F) or three (3F) features. Chance level (RMSE) is 7.447. All results on test data and averaged over 10 runs. (Left) Error using training graphs that code only color. (Right) Error using training graphs that simultaneously code gender and color.

The effect of coding gender and color simultaneously on gender estimation is shown in Table 2, right (compare to Table 2, left). The  $\text{ELL}^{g,c}\text{-}L$  graphs yield higher accuracy than the  $\text{ELL}^g\text{-}L$  graphs. The results on color estimation using the  $\text{ELL}^{g,c}\text{-}L$  graphs are shown in Table 3, right (compare to Table 3, left). The slowest feature extracted represents mostly gender. However, it also contains color information because it allows color estimation better than the chance level. When 3 features are preserved, the  $\text{ELL}^{g,c}\text{-}L$  graphs yield higher accuracy than the  $\text{ELL}^c\text{-}L$  graphs. Similar experimental results have been reported, e.g. by Guo and Mu (2014), who have shown that age estimation improves when gender and race labels are also considered.

**Learning label transformations.** We verify that the method can learn other labels implicitly described by the data. More precisely, we use GSFA to learn labels  $(\ell_1)^2$  and  $(\ell_1)^3$ , which are distorted versions of the original gender label  $\ell_1$ . The graphs constructed for this purpose are denoted  $\text{ELL}^{g-40(\ell_1)^2}$  and  $\text{ELL}^{g-40(\ell_1)^3}$ , respectively. Both of them include 39 auxiliary labels besides the main distorted label. To better approximate the target labels, more complex nonlinearities are used in some of the nodes of the hierarchical networks. The  $(\ell_1)^2$  network is identical to the  $\ell_1$  network, except that in the top node the quadratic expansion is used instead of the 0.8Expo expansion. Similarly, the  $(\ell_1)^3$  network uses the quadratic expansion in the 7th layer, and the 6th-degree polynomial expansion in the top node. In both networks, the output dimension of the node in the 7th layer is set to 3 to avoid overfitting due to the expansion in the 8th layer.

The corresponding label estimations are shown in Figure 6. For comparison, also the  $\text{ELL}^g\text{-40}$  graph is included. The results prove that the ELL method can also be used to learn distortions of the main label. Admittedly, the accuracy of the estimations (normalized by the respective chance levels) decreases even though we increase the complexity of the feature space.

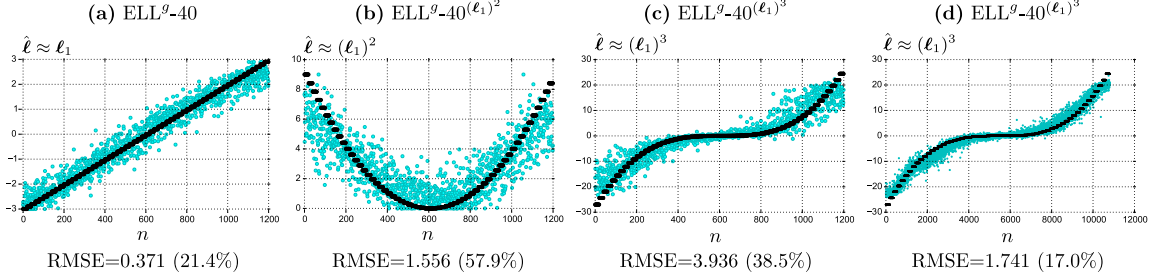


Figure 6: Plots (a) to (c) show the label estimations on test data when different distorted versions of  $\ell_1$  are learned. The linear scaling mapping was used. Therefore, the estimations are only generated from the slowest feature. Ground-truth values shown in thicker black. The RMSE is expressed in brackets as a percentage of the chance level. Plot (d) is analogous to (c) but shows training data.

## 4.2 Analysis of Pre-Defined Training Graphs

In this section, we use the method of Section 3.1 to extract the optimal free responses of three graphs (reordering, serial and ELL-4). The optimal free responses and their  $\Delta$  values (alternatively, the eigenvectors  $\mathbf{u}_j$  and eigenvalues  $\lambda_j$ ) fully characterize the properties of a training graph, and provide another representation of it that might be more useful in some scenarios.

We compute optimal free responses using (33)–(35) and their delta values using (39). Therefore, these results have been obtained analytically. We plot them in Figure 7, which shows an arbitrary label to be learned (top), and three different graphs that can be used for this purpose. Only  $N = 30$  samples (ordered by increasing label) were used to ease visualization, but the plots behave similarly for larger  $N$ . The employed graphs are as follows. The reordering graph has been extended with two edge weights  $\gamma_{0,0} = 1$  and  $\gamma_{N-1,N-1} = 1$  to fulfill the consistency restriction (13), which is required by the method. These weights introduce a constant scaling  $N/(N+2)$  of the delta values, without any further consequence. The serial graph (Section 2.3) has  $K = 15$  groups of 2 samples each. The ELL-4 graph (Sections 3.2–3.4) is constructed with the original labels  $\ell_1(n) = \ell(n)$ , and 3 auxiliary labels computed using (67).

The figure shows that the most remarkable difference between the graphs is the number of optimal free responses with  $\Delta < 2.0$ , which is 14 for the reordering graph, 6 for the serial graph, and 4 for the ELL-4 graph, for the parameters above. For arbitrary parameters, the reordering, serial and ELL- $L$  graphs have  $\lfloor (N-1)/2 \rfloor$ ,  $\lfloor (K-1)/2 \rfloor$ , and, depending on the eigenvalues, up to  $L \leq N-1$  optimal free responses with  $\Delta < 2.0$ , respectively.

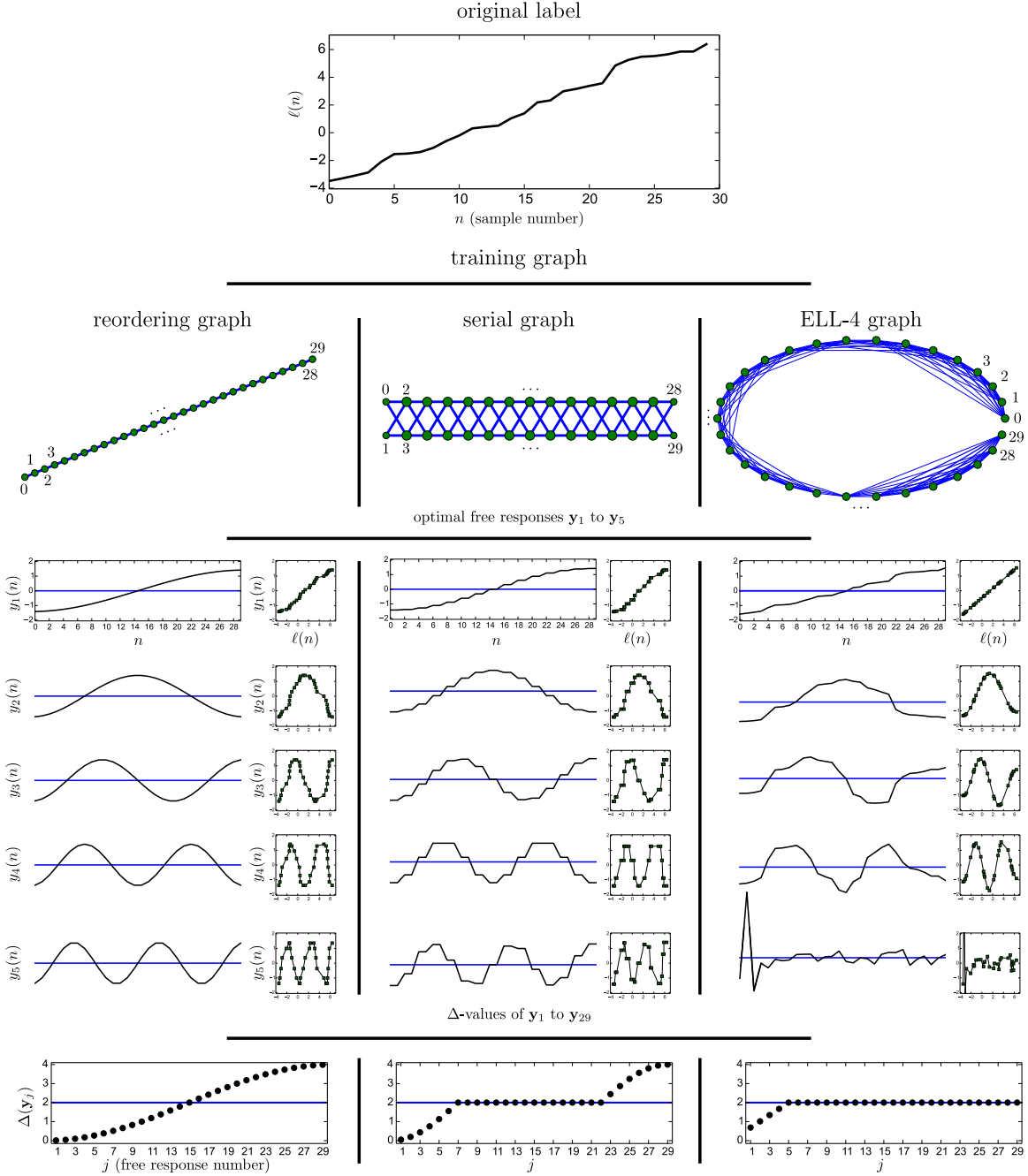


Figure 7: An arbitrary label (top) and three graphs that can be used to learn it. The five slowest optimal free responses of each graph are plotted, as well as the delta values of all optimal free responses. The ELL-4 graph is almost fully connected, but here only the strongest 30% of the connections are displayed. Samples have an index  $n$  from 0 to 29, and free responses have an index  $j$  from 1 to 29. The free responses are also plotted against the original label. The polarity of the free responses was adjusted once to make them negative for the first sample.

Although the graphs differ considerably in their connectivity, their first four to five optimal free responses have a somewhat similar shape. Since in all graphs the slowest free response  $\mathbf{y}_1$  is increasing, a monotonic mapping would be enough to approximate the label for any of them. However, for the serial graph the slowest response is constant within each group, which might lower accuracy due to a discretization error. The ELL-4 graph has been tailored to learn a particular label, and therefore  $\mathbf{y}_1$  is exactly  $\ell_1$  (the original label) except for an offset and scaling.

The analysis makes clear that the serial and ELL-4 graphs are *more selective* than the reordering graph regarding the features that they consider slow. To illustrate why this might be an advantage, consider a scaled and noisy version  $\hat{\mathbf{y}}_1$  of  $\ell_1$ . More concretely,  $\hat{y}_1(n) = \frac{\sqrt{2}}{2}\ell_1(n) + \frac{\sqrt{2}}{2}e(n)$ , where  $e(n)$  is an i.i.d. zero-mean unit-variance noise signal. When the reordering graph is used, the feature  $\hat{\mathbf{y}}_1$  has an average  $\Delta$ -value of about 1 (i.e.  $\langle \Delta_{\hat{\mathbf{y}}_1} \rangle \approx 1$ ), and therefore such a feature would appear to be faster than the auxiliary feature  $\mathbf{y}_6$ , because  $\Delta_{\mathbf{y}_6} \approx 0.38$ . Hence, a GSFA node trained with the reordering graph would favor the extraction of  $\mathbf{y}_6$  over  $\hat{\mathbf{y}}_1(n)$ , even though  $\hat{y}_1(n)$  is more similar to the label. In contrast, the serial and ELL-4 graphs might favor the extraction of  $\hat{\mathbf{y}}_1$ , because for these graphs  $\Delta_{\mathbf{y}_6}$  is close to 2.0.

### 4.3 Compact Discriminative Features for Classification

A well-known algorithm for supervised dimensionality reduction for classification is Fisher discriminant analysis (FDA). According to the theory of FDA, if there are  $C$  classes,  $C - 1$  features define a  $C - 1$  dimensional subspace that best separates the classes. In practice, one typically uses all these  $C - 1$  features, because all of them contain discriminative information and contribute to classification accuracy. The same holds for GSFA if the clustered training graph is used (GSFA+clustered), because in this case the features learned are equivalent to those of FDA (Klampfl and Maass, 2010; Escalante-B. and Wiskott, 2013).

One can take advantage of hierarchical processing to do classification using the clustered graph (HGSFA+clustered). However, when the number of classes  $C$  is large (e.g.  $C \geq 100$ ) it might become expensive to preserve  $C - 1$  features in each node, because the size of the input to subsequent nodes would be a multiple of  $C - 1$ . This dimensionality would be further increased by the expansion function, resulting in a large training complexity. For instance, consider a 3-node nonlinear network for classification with two GSFA nodes in the first layer and one in the top. Suppose the first two nodes have output dimensionality  $C - 1 = 99$  so that the input into the top node would be 198, and suppose that the top node applies a quadratic expansion to its input data before linear GSFA. The expanded data would have dimensionality  $I' = 19,701$ . The combination of a large sample dimensionality  $I'$  and a large number of samples  $N$  (with  $N \gg I'$  to avoid overfitting) would result in considerable computational and memory costs. Therefore, if we could code the class information in the first layer more compactly, we could reduce the output dimensionality of the first-layer nodes and reduce overfitting, aiming at increasing classification accuracy.

In this section, we use the theory of explicit learning with multiple labels to compute compact features for classification using GSFA. We classify images of  $C = 32$  traffic signs from the German traffic sign recognition benchmark database (Houben et al., 2013).

The images are represented as  $48 \times 48$ -pixel color (RGB) images (see Figure 8). We use only 32 out of 43 traffic signs with the most samples. For the training data, we use the same number of samples for each class (traffic sign), namely 2,160 of them, making a total of 69,120 images. To reach 2,160 samples per class, images of some classes are used up to 6 times (since the database is unbalanced). The images used for training are distorted by a random rotation  $r$  of  $-3.15 \leq r \leq 3.15$  degrees, horizontal and vertical translations  $\Delta_x, \Delta_y$  with  $-1.73 \leq \Delta_x, \Delta_y \leq 1.73$  pixels, and a scaling factor  $s$  with  $0.91 \leq s \leq 1.09$ . The purpose of the distortion is to improve generalization and provide invariances to small misalignments. We use the official test data, which ensures that the images originate from physical signs different from the ones used for training. The test data consists of 9,030 undistorted images.



Figure 8: The 32 traffic signs learned, one image per class.

We used a simple (non-hierarchical) GSFA architecture, in which PCA is applied first to reduce the dimensionality to 120 principal components. Afterwards, quadratic GSFA is applied using different training graphs, described below. Finally, since this is a classification problem, a nearest centroid classifier is used instead of linear scaling.

The ELL method is used to construct two training graphs with binary target labels (i.e., a label is either 1 or  $-1$ ). The first one has 5 labels (compact+5) and the second one has 31 (compact+31). The target labels are defined in Table 4. Notice that the first 5 labels (for both graphs) suffice, in principle, to fully code the class information.

For the compact+5 graph, identical eigenvalues ( $\lambda_1^1 = \lambda_2^1 = \lambda_3^1 = \lambda_4^1 = \lambda_5^1 = 0.2$ ) are used to express equal importance of the target labels. The compact+31 graph has been included to show the effect of auxiliary labels  $\ell_6, \ell_7, \dots, \ell_{31}$ . For this graph, the first five eigenvalues ( $\lambda_1^2, \lambda_2^2, \dots, \lambda_5^2 = (0.056, 0.056, \dots, 0.056)$ ) are identical, but the rest decrease linearly: ( $\lambda_6^2, \lambda_7^2, \dots, \lambda_{31}^2 = (0.053, 0.051, \dots, 0.002)$ ), where only three decimal places are shown. Thus, the importance given to the auxiliary labels decreases from  $\ell_6$  to  $\ell_{31}$ . For both graphs, we scale the eigenvalues to make their sum equal to 1.

We choose  $C = 2^5$  classes, because powers of two make it simple to obtain binary labels with a weighted zero mean, weighted unit variance, and weighted decorrelation, as follows. The first five original labels can be computed as  $\ell_j(c) = 2(\frac{c-1}{2^5-j} \bmod 2) - 1$ , where the

$c \rightarrow$	1	2	3	4	5	6	7	8	9	...	16	17	...	30	31	32
$\ell_1(c)$	-1	-1	-1	-1	-1	-1	-1	-1	-1	...	-1	1	...	1	1	1
$\ell_2(c)$	-1	-1	-1	-1	-1	-1	-1	-1	1	...	1	-1	...	1	1	1
$\ell_3(c)$	-1	-1	-1	-1	1	1	1	1	-1	...	1	-1	...	1	1	1
$\ell_4(c)$	-1	-1	1	1	-1	-1	1	1	-1	...	1	-1	...	-1	1	1
$\ell_5(c)$	-1	1	-1	1	-1	1	-1	1	-1	...	1	-1	...	1	-1	1
$\ell_6(c)$	-1	1	1	-1	1	-1	-1	1	1	...	-1	1	...	-1	-1	1
$\ell_7(c)$	-1	-1	1	1	1	1	-1	-1	1	...	1	1	...	1	-1	-1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$\ell_{30}(c)$	-1	1	-1	1	1	-1	1	-1	-1	...	-1	-1	...	-1	1	-1
$\ell_{31}(c)$	-1	1	1	-1	-1	1	1	-1	-1	...	-1	-1	...	1	1	-1

Table 4: Target labels used to code the class information, compactly expressed as a function of the class number  $c$ . The compact+5 graph is constructed with labels  $\ell_1$  to  $\ell_5$ , whereas the compact+31 graph with  $\ell_1$  to  $\ell_{31}$ . The first five labels can be seen as the original ones and the rest as auxiliary.

division is integer division and “mod” is the modulo operation. The auxiliary labels are computed as the product of two or more labels  $\ell_1$  to  $\ell_5$ , possibly multiplied by a factor  $-1$  to make the label assigned to the first class negative. More concretely,  $\ell_6$  is the product of all original labels,  $\ell_7$  to  $\ell_{11}$  are all the products of four of them,  $\ell_{12}$  to  $\ell_{21}$  are all the products of three, and  $\ell_{22}$  to  $\ell_{31}$  are all the products of two (e.g.,  $\ell_6 = \ell_1\ell_2\ell_3\ell_4\ell_5$ ,  $\ell_7 = -\ell_1\ell_2\ell_3\ell_4$ ,  $\ell_8 = -\ell_1\ell_2\ell_3\ell_5$ ,  $\ell_{30} = -\ell_3\ell_5$ ,  $\ell_{31} = -\ell_4\ell_5$ ).

For both graphs, we set  $\mathbf{v} = \mathbf{1}$ . The corresponding eigenvectors are  $\mathbf{u}_j \stackrel{(43)}{=} Q^{-1/2}\ell_j$ , where  $Q \stackrel{(5)}{=} N \cdot 1 = 69,120$  ( $N$  is the number of training images). These eigenvectors are also binary and allow for a fast computation of the covariance matrix in  $\mathcal{O}(LNI^2 + I^3)$  operations, where  $L$  is the number of target labels.

The classification error is plotted in Figure 9, where the number of slow features  $d$  given to a nearest centroid classifier ranges from 4 to 31. For comparison, the clustered graph is also evaluated. For  $d = 5$  features, the compact+5 graph results in the best accuracy with an error rate of 11.67%, against 12.42% (compact+31) and 29.74% (clustered). However, the error rate of the compact+5 graph increases if one preserves more than 5 features, indicating that additional features contain little or no discriminative information. For  $6 \leq d \leq 30$ , the compact+31 graph yields clearly better accuracy than the other graphs. Interestingly, for  $d = 31 = C - 1$  features, the compact+31 and clustered graph give identical error rates of 2.89%, which is their top performance. In this case, the features extracted are *different* but contain the *same information* since they can be mapped to each other linearly. In other words, the first 31 free responses of both graphs describe the same subspace. Any *single* slow feature from the compact+31 graph contains ideally 1 bit of discriminative information (which might be redundant to the others). In contrast, the first features extracted by the clustered graph might sacrifice discriminative information to minimize within-class variance (e.g., a feature  $\mathbf{y}(c) = ((\frac{C}{2})^{1/2}, -(\frac{C}{2})^{1/2}, 0, 0, \dots, 0)$  has minimal (zero) within-class variance but provides little discriminative information, because only the first two classes can be identified from it). Using  $d > 31$  features does not improve accuracy in any case.



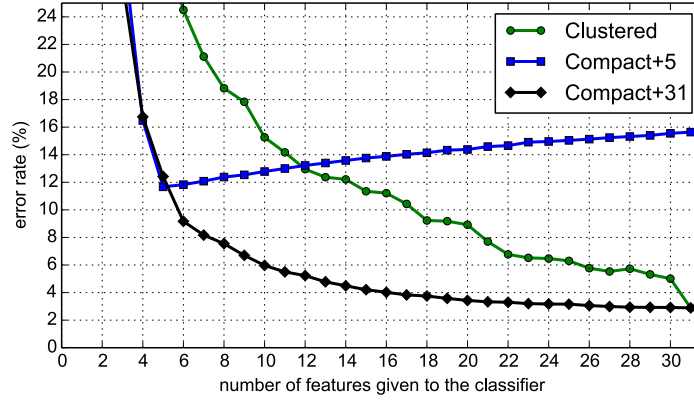


Figure 9: Classification error when GSFA is trained with the compact+5, compact+31 and clustered graph (FDA). This error is a function of the graph and the number of slow features  $d$  passed to the classifier. For the clustered graph, dropping even a single feature might increase the error rate significantly. For instance, the error rate of using 30 features computed with the clustered graph is worse than the error rate of 13 features computed with the compact+31 graph. Performance on 9,030 test samples, averaged over 10 runs (largest standard deviation 0.15%).

One assumption of this method is that the feature space is complex enough to allow the extraction of features that approximate the binary labels. If the feature space is poor, the compact graphs might not bring any advantage over the clustered one.

The results suggest that if the number of features to be preserved  $d < C - 1$  is known, one might improve accuracy by creating a graph that uses exactly  $d$  target labels. If the number of output features is unclear, a graph coding several auxiliary labels with decreasing eigenvalues, e.g. compact+ $(C - 1)$ , might provide better robustness (see Figure 9).

## 5. Discussion

In this article, we propose exact label learning (ELL) for the construction of training graphs, in which the final label estimation is just a linear transformation of the slowest feature extracted. The method allows the direct solution of regression problems with GSFA, without having to recur to a supervised post-processing step. In other words, given a new input sample (e.g. an input image) the first feature computed using an ELL graph directly provides an approximation of the label (or a linear transformation of it). In practice, even better results may be achieved using more than one feature and supervised post-processing.

It is crucial to emphasize that GSFA optimizes feature slowness, which depends on the particular training graph used, and not label estimation accuracy. However, when the ELL method is used, the training graphs define a slowness objective that requires optimizing an output similarity function, where the similarities are intimately related to the desired label similarities. As a result, the feature slowness and estimation accuracy objectives become equivalent when  $\mathcal{F}$  becomes unlimited. That is, the slowest possible features that can be

extracted (i.e. optimal free responses) are equal to a normalized version of the label(s). In practice,  $\mathcal{F}$  is finite to allow generalization from training to test data and, if the features extracted are slow enough (i.e. close to the optimal free responses), they are also good solutions to the original regression problem.

Supervised learning problems on high-dimensional data are of great practical importance, but they frequently result in systems with large computational demands. A common approach is to apply feature extraction, dimensionality reduction, and a supervised learning algorithm. A promising alternative approach is hierarchical GSFA (HGSFA), because its complexity scales in some cases even linearly w.r.t. the input dimensionality and the number of samples. Furthermore, when HGSFA is trained with an ELL graph, the resulting architecture is simple and homogeneous, as shown in Figure 1.c.

We have proved the usefulness of the ELL method by showing three types of applications that are relevant in practice: ELL regression with multiple labels, analysis of training graphs, and classification with compact discriminative features.

The results show that encoding auxiliary labels derived from the original one (e.g. “higher-frequency” transformations of it) improve performance. This is particularly relevant for cascaded or convergent hierarchical GSFA networks. The auxiliary labels contain information that (partially) determines the original label. A GSFA node that receives this information might be able to extract the original label more accurately than without it.

Multiple labels can be learned simultaneously, for instance to code different aspects of the input at once (e.g. object color, size, shape, orientation). The use of multiple labels is inspired by biological systems, where complementary information channels have been observed and might improve feature robustness, for example under incomplete information (Krüger et al., 2013). Learning gender and color simultaneously yielded clearly smaller estimation errors than when gender and color were estimated separately. This shows that multiple label learning is not only theoretically possible, but that coding complementary information channels might boost accuracy in practice. For instance, an automatic system for face processing might benefit from the simultaneous extraction of the subject’s identity, age, gender, race, pose, and expression.

The experiments on gender (and skin-color) estimation from artificial face images verify that the ELL method works in practice. The complexity of training a single GSFA node with an ELL graph is  $\mathcal{O}(IN^2 + I^2N + I^3)$  operations, where  $I$  is the input dimensionality (possibly after a nonlinear expansion), and  $N > I$  is the number of samples. For comparison, the serial graph has a complexity of  $\mathcal{O}(I^2N + I^3)$ . Thus, the main limitation of using ELL graphs is the training complexity.

The analytical and practical results show the strength of the serial graph when only a single label is available. In this case, the ELL graph provided marginally better estimations than the serial one (an RMSE of 0.345 with the ELL<sup>g</sup>-40 graph vs. 0.349 with the serial graph, in both cases using 3 features and the soft GC post-processing method), but the computation time was 25 times larger. Although the shape of the slowest feature extracted with the serial graph might be less similar to the label, a monotonic transformation of the slowest feature learned by a nonlinear supervised step (e.g. soft GC) might suffice to approximate it.

However, the results show that if two or more (intrinsically connected) labels are available, the accuracy of using ELL graphs may further increase. Efficient pre-defined graphs

are not available in this case. In the gender estimation experiment, the RMSE was improved to 0.277 by jointly learning gender and skin ( $\text{ELL}^{g,c}$ -( $2 \times 5$ ) graph, 3 features, soft GC). Hence, a particularly promising application for the ELL method is multiple label learning.

Various methods for mapping the slowest feature to a label were tested. The linear scaling method is interesting from a theoretical point of view. However, as one would expect, it provided worse accuracy for test data than the soft GC method, which is nonlinear and supervised. Therefore, the latter might be preferred in practical applications. Moreover, in this scenario, supervised post-processing methods might be computationally inexpensive, because their input is frequently low-dimensional (e.g., we used 1 to 3 slow features for gender estimation).

Although ELL was originally designed for regression, we show that it can also be useful for classification when particular labels are learned. The experiment on traffic sign classification shows the benefit of using compact discriminative features, implemented here by learning multiple binary labels. The resulting system has a much smaller classification error than the clustered graph (equivalent to nonlinear FDA) when the number of output dimensions is less than  $C - 1$ , where  $C$  is the number of classes. One can combine the method with HGSFA for classification with many classes, coding the discriminative information in much fewer than  $C - 1$  features and reducing the number of signals computed in the network, which might also reduce overfitting. Although ideally  $\log_2(C)$  binary target labels suffice for perfect classification, the experiments show that additional target labels via auxiliary labels improve classification accuracy in practice.

Interestingly, the clustered graph for  $C$  classes (equivalent to FDA) and the compact+( $C - 1$ ) graph are equivalent if the latter is constructed with constant positive eigenvalues  $\lambda_1 = \dots = \lambda_{C-1} = 1/(C - 1)$ . The reason for this equivalence is that this compact+( $C - 1$ ) graph would only have within-class transitions, because transitions between different classes cancel out each other. Therefore, the clustered graph can be seen as a special case of the compact+( $C - 1$ ) graph, with maximum label redundancy ( $C - 1$  target labels) and giving equal importance (eigenvalues) to all of them.

Future work in the scope of ELL might include the construction of training graphs that are efficient (like the serial one) but at the same time offer the versatility of the ELL graphs. For example, up to now, efficient (pre-defined) training graphs can only learn a single target label (and its higher-frequency harmonics). Therefore, we are interested in designing *efficient* graphs that allow learning multiple labels.

Another possible application of ELL is the generation of (partially) sparse feature representations. For instance, for  $C = 8$  classes, one might learn features  $\ell_1(c) = (2, -2, 0, 0, 0, 0, 0, 0)$ ,  $\ell_2(c) = (0, 0, 2, -2, 0, 0, 0, 0)$ ,  $\ell_3(c) = (0, 0, 0, 0, 0, 0, 2, -2)$ ,  $\ell_4(c) = (0, 0, 0, 0, 0, 0, 2, -2)$ ,  $\ell_5(c) = (2^{1/2}, 2^{1/2}, -2^{1/2}, -2^{1/2}, 0, 0, 0, 0)$ ,  $\ell_6(c) = (0, 0, 0, 0, 2^{1/2}, 2^{1/2}, -2^{1/2}, -2^{1/2})$ , and  $\ell_7(c) = (1, 1, 1, 1, -1, -1, -1, -1)$ . These features have zero mean, unit variance, and are decorrelated. Furthermore,  $\ell_1$  to  $\ell_4$  make the smallest possible contribution to the  $L_1$  norm, thus this construction is greedy (it might not minimize the  $L_1$  norm globally).

An open research question is how to choose the auxiliary labels. It is clear they should depend on the original label(s), but it is unclear how to compute them to maximize estimation accuracy, how many of them should be coded, and which eigenvalues should be used. For classification with  $C = 32$  classes, *linearly* decreasing eigenvalues (for the auxiliary labels) provided great results, but other eigenvalues might be better if  $C$  is very large.

Hierarchical processing and the slowness principle are two powerful brain-inspired learning principles. The strength of SFA originates from its theoretical foundations in the field of learning of invariances and the generality of the slowness principle. For practical supervised learning applications, HGSFA provides good accuracy and efficiency and still profits from strong theoretical foundations. An advantage of relying on such general principles is that the resulting algorithms are application independent and not confined to a particular problem or input feature representation. Of course, fine tuning the network parameters and the integration of problem-specific knowledge are always possible for additional performance. The proposed method explores the limits of HGSFA and is valuable as a theoretical tool for the analysis and design of training graphs. However, the results show that with certain adaptations (e.g. the use of supervised post-processing) it is also sufficiently robust to be applied to practical computer vision and machine learning tasks.

## References

- M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Comput.*, 15(6):1373–1396, June 2003. ISSN 0899-7667.
- P. Berkes. Pattern recognition with Slow Feature Analysis. Cognitive Sciences EPrint Archive (CogPrints), February 2005a. URL <http://cogprints.org/4104/>.
- P. Berkes. Handwritten digit recognition with nonlinear Fisher discriminant analysis. In *ICANN*, volume 3697 of *LNC*S, pages 285–287. Springer Berlin/Heidelberg, 2005b.
- A. N. Escalante-B. and L. Wiskott. Heuristic evaluation of expansions for non-linear hierarchical Slow Feature Analysis. In *Proc. The 10th International Conference on Machine Learning and Applications*, pages 133–138, Los Alamitos, CA, USA, 2011.
- A. N. Escalante-B. and L. Wiskott. How to solve classification and regression problems on high-dimensional data with a supervised extension of Slow Feature Analysis. *Journal of Machine Learning Research*, 14:3683–3719, December 2013.
- P. Földiák. Learning invariance from transformation sequences. *Neural Computation*, 3(2):194–200, 1991.
- G. Guo and G. Mu. A framework for joint estimation of age, gender and ethnicity on a large database. *Image Vision Comput.*, 32(10):761–770, 2014.
- X. He and P. Niyogi. Locality Preserving Projections. In *Neural Information Processing Systems*, volume 16, pages 153–160, 2003.
- G. E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40(1-3):185–234, 1989.
- S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel. Detection of traffic signs in real-world images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, number 1288, 2013.
- S. Klampfl and W. Maass. Replacing supervised classification learning by Slow Feature Analysis in spiking neural networks. In *Proc. of NIPS 2009: Advances in Neural Information Processing Systems*, volume 22, pages 988–996. MIT Press, 2010.

- N. Krüger, P. Janssen, S. Kalkan, M. Lappe, A. Leonardis, J. Piater, A. Rodriguez-Sanchez, and L. Wiskott. Deep hierarchies in the primate visual cortex: What can we learn for computer vision? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1847–1871, 2013.
- G. Mitchison. Removing time variation with the anti-Hebbian differential synapse. *Neural Computation*, 3(3):312–320, 1991.
- L. Wiskott. Learning invariance manifolds. In *Proc. of 5th Joint Symposium on Neural Computation, San Diego, CA, USA*, volume 8, pages 196–203. Univ. of California, 1998.
- L. Wiskott. Slow Feature Analysis: A theoretical analysis of optimal free responses. *Neural Computation*, 15(9):2147–2177, 2003.
- L. Wiskott and T. Sejnowski. Slow Feature Analysis: Unsupervised learning of invariances. *Neural Computation*, 14(4):715–770, 2002.